REMARKS/ARGUMENTS

Claims 1-19 are pending in the present application. Reconsideration of the claims is respectfully requested.

I. Examiner Interview

Tentatively scheduled for 3/12/07 at 10AM Central. Examiner has not yet responded with a confirmation of the interview schedule or an alternate schedule. We will continue our attempts to contact the Examiner until the due date of this response.

II. Objection, Information Disclosure Statement

An English translation of the abstract of reference JP10260820A, a Japanese patent application publication, is included with the present response. This was first identified in the IDS filed with the present application on July 24, 2003. Applicants request the Examiner to consider the reference.

III. 35 U.S.C. § 102, Anticipation

The Examiner has rejected claims 1-19 under 35 U.S.C. § 102(b) as being anticipated by *Donohue et al.*, Method and a Mechanism for Synchronized Updating of Interoperating Software, U.S. Patent No. 6,202,207 (March 13, 2001) (hereinafter "*Donohue*"). This rejection is respectfully traversed.

The Examiner has rejected these claims stating:

> Claims 1-19 are rejected under 35 U.S.C. 102(b) as being anticipated by Donohue (US 6,202,207 BI). Donohue further teaches a method for testing the compatibility of software versions (see at least FIGS.4A-B & associated text), the method comprising the computer implemented steps of:
>
> responsive to an installation of a new software module in a data processing system, performing an inventory on an existing set of software modules resident in the data processing system (see at least 10, 80, 90 FIG.1 & associated text; 200, 21 0, 230, 290, 41 0 FIG.4A & associated text);
>
> referring to a knowledge base of versions of respective software modules to obtain compatibility information for the new software module with the existing set of software modules (see at least 40 FIG. I & associated text; 110, 120, 130 FIG.2 & associated text; 250, 260 FIG.4A & associated text); and
>
> providing the compatibility information from the knowledge base, wherein the compatibility information is used to determine whether to install the new software module (see at least 310 FIG.4B & associated text).

Office Action dated December 19, 2006, pp. 2-3.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.,* 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

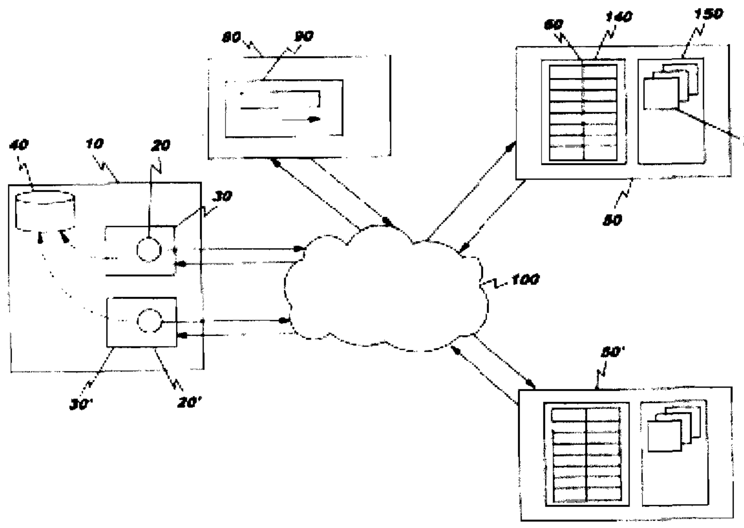
Claim 1 recites:

> A method for testing the compatibility of software versions, the method comprising the computer implemented steps of:
>
> responsive to an installation of a new software module in a data processing system, performing an inventory on an existing set of software modules resident in the data processing system;
>
> referring to a knowledge base of versions of respective software modules to obtain compatibility information for the new software module with the existing set of software modules; and
>
> providing the compatibility information from the knowledge base, wherein the compatibility information is used to determine whether to install the new software module.

In the present case, each and every step in claim 1 is not shown in the cited reference. In particular, the steps of performing, referring, and providing are not shown in *Donohue* as recited in claim 1.

The Examiner cites the following figures references and corresponding text in *Donohue* as teaching the performing step:

FIG. 1



Donohue, Figure 1.

The Examiner incorrectly believes that reference numerals 10, 80, and 90 in this figure teach the performing step as recited in claim 1. In this figure, *Donohue* presents a schematic representation of a computer network. A computer system in the network contains an installed update component, servers in the network contains a listing of available updates and software resources for applying the updates, and a search engine for locating the servers with the desired update. The updates in this figure are updates for software applications that a specific update component can access and update.

Reference numeral 10 is a computer system in this figure. *Donohue* provides:

> As shown in FIG. **1**, an updater component **20** is installed in system memory of a conventional network-connected computer system **10**, together with an associated computer program **30**.

Donohue, col.7, ll. 54-57.

Donohue's disclosure supports the above description of *Donohue's* reference numeral 10 in Figure 1. Reference numeral 10 is a computer system in this figure. Of course, a computer system can be programmed to perform any conceivable task, but for the purposes of anticipation under 35 U.S.C. § 102(b), depicting a computer system does not teach a particular step in a particular process that may or may not be encoded into a software application. Nowhere does Donohue teach that network-connected computer system **10** executes the performing step of claim 1. Here, reference numeral 10 and its corresponding description is insufficient to teach the performing step as recited in claim 1.

Reference numeral 80 is a server system and reference numeral 90 is a search engine. *Donohue* provides:

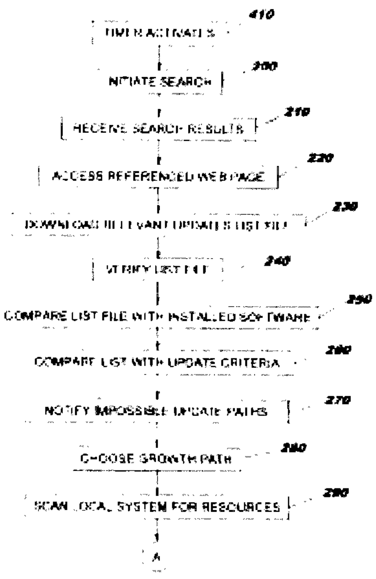
> Also, shown within the network **100** is a server system **80** on which a search engine **90** is installed for use in finding update source locations on the network.

Donohue, col. 10, ll. 5-8.

Similar to the reasoning as to the computer system 10, above, a server or a search engine depicted in a figure cannot teach a particular step in a particular process that may or may not be encoded into a software application. Here, reference numerals 80 and 90 and their corresponding descriptions are insufficient to teach the performing step as recited in claim 1. Specifically, neither of the reference numerals cited by the Examiner teaches performing an inventory on an existing set of software modules resident in a data processing system. Applicants stipulate that the cited reference numerals may teach a data processing system. However, the "performing an inventory", "an inventory", "an inventory on an existing set of software modules" are all absent in the teachings of the citations made by the Examiner.

The Examiner further cites to the following reference numerals in figure 4A of *Donohue* as teaching the performing step:

FIG. 4A



410
TIMER ACTIVATES
200
INITIATE SEARCH
210
RECEIVE SEARCH RESULTS
220
ACCESS REFERENCED WEB PAGE
230
DOWNLOAD RELEVANT UPDATES LIST FILE
240
VERIFY LIST FILE
250
COMPARE LIST FILE WITH INSTALLED SOFTWARE
260
COMPARE LIST WITH UPDATE CRITERIA
270
NOTIFY IMPOSSIBLE UPDATE PATHS
280
CHOOSE GROWTH PATH
290
SCAN LOCAL SYSTEM FOR RESOURCES
A

Donohue, Figure 4A.

The Examiner incorrectly believes that reference numerals 200, 210, 230, 290, and 410 in this figure teach the performing step as recited in claim 1. In this figure, *Donohue* presents a sequence of operation of *Donohue's* update process using *Donohue's* updater component. The steps cited by the Examiner are described in *Donohue* as follows:

> The operation of an updater component will now be described, with reference to FIGS. **3** and **4.** When an installed updater component executes, in response to completion of a cycle period or in response to a request from another software product's updater component, its first action is to initiate **200** a search for available updates to the particular software product.
>
> A URL identifying the relevant Web site **140** for update information is returned **210** to the updater component as a result of the search.
>
> The updater component uses the URL to access **220** the list **60** and downloads **230** a file **160** comprising the portion of the list **60** of available updates which relates to the particular product.
>
> The updater component performs **290** (see FIGS. **3** and **4**) a scan of the operating system file system to check whether the required software resources are already available on the local computer system.
>
> The updater generates **380** a report and writes **390** to log records, and then quits execution **400** (in the preferred embodiment the updater goes into a sleep or idle state) until activated again **410** upon expiry of a predetermined update cycle period.

Donohue, col. 10, ll. 16-22, 31-33, 39-42; col. 11, ll. 46-49; col. 13, ll. 11-15.

According to the above descriptions in *Donohue*, step 200 in *Donohue's* Figure 4A is a step for initiating search for updates for a software. Step 210 is returning update information in response to the search. Step 230 is downloading a file from a list of updates that results from the search. Step 290 is scanning the computer's file system by the updater component to determine if the software resources needed for the update are available thereon. Finally, step 410 is reactivating the updater after a predetermined time has elapsed. Therefore, *Donohue* fails to teach the performing step as recited in claim 1. These teachings in *Donohue* do not teach at least the "performing an inventory", "an inventory", "an inventory on an existing set of software modules" as recited in claim 1.

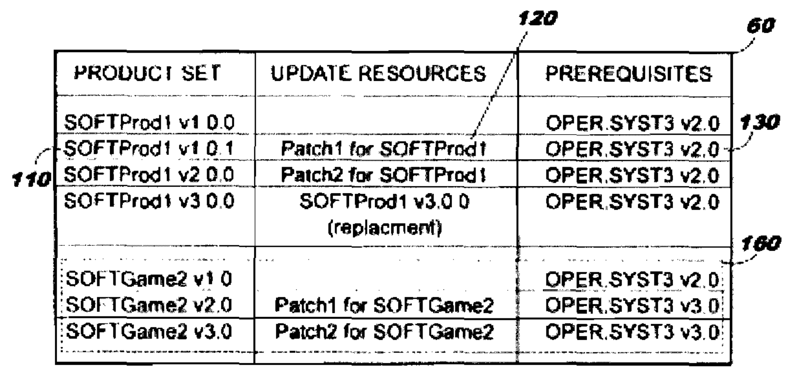
Next, the Examiner cites reference numeral 40 in Figure 1, reproduced above, as teaching the referring step of claim 1. *Donohue* describes reference numeral 40 as follows:

> It is a feature of the preferred embodiment of the invention that each updater component can locate, can be located by, and can communicate with other updater components which manage other software products. This capability is used when one updater component requires another one to update to a specific level before the former can execute its own update, as will be discussed below, This is enabled by the updater components registering within the operating system or other repository 40.

Donohue, col. 8, ll. 10-18.

According to the above description, several updaters can exist on the various data processing systems depicted in the network of *Donohue* Figure 1. Accordingly, *Donohue* reference numeral 40 is a repository <u>for updaters to locate each other</u> in the computer network of Figure 1. A repository of one information cannot teach a repository of another information for the purposes of anticipation under 35 U.S.C. § 102(b). Thus, a repository containing information about the updaters does not teach "a knowledge base <u>of versions of respective software modules to obtain compatibility information</u>" as recited in claim 1.

The Examiner additionally cites the following figures references and corresponding text in *Donohue* as teaching the referring step:



120
60
130
110
160

PRODUCT SET	UPDATE RESOURCES	PREREQUISITES
SOFTProd1 v1 0.0		OPER.SYST3 v2.0
SOFTProd1 v1 0.1	Patch1 for SOFTProd1	OPER.SYST3 v2.0
SOFTProd1 v2 0.0	Patch2 for SOFTProd1	OPER.SYST3 v2.0
SOFTProd1 v3 0.0	SOFTProd1 v3.0 0 (replacment)	OPER.SYST3 v2.0
SOFTGame2 v1 0		OPER.SYST3 v2.0
SOFTGame2 v2.0	Patch1 for SOFTGame2	OPER.SYST3 v3.0
SOFTGame2 v3.0	Patch2 for SOFTGame2	OPER.SYST3 v3.0

Donohue, Figure 2.

The Examiner incorrectly believes that reference numerals 110, 120, and 130 in this figure teach the performing step as recited in claim 1. In this figure, *Donohue* presents a software vendor provided list of software versions, including the resources needed to update to a certain version, and any prerequisites for updating to that version. The steps cited by the Examiner are described in *Donohue* as follows:

> The entries in the software updates list **60** include for each software product version **110** an identification **120** of the software resources required for applying the update and an identification **130** of its prerequisite software products and their version numbers.

Donohue, col. 9, ll. 59-63.

According to the above descriptions in *Donohue*, reference numeral 110 is a software version for each software product. Ref num 120 is an identification of resources needed to get to the software version 110. Reference numeral 130 is an identification of prerequisites for updating to the software version 110. These teachings in *Donohue* do not teach at least the "compatibility information" as recited in the referring step of claim 1. Specifically, in this step, the compatibility information is for the new software module with the existing set of software modules. Neither of the cited reference numerals and their corresponding descriptions teaches compatibility information about new software and existing set of software as claimed.

The Examiner further cites reference numerals 250 and 260 in Figure 4A, reproduced above, as teaching the referring step of claim 1. *Donohue* describes reference numerals 250 and 260 as follows:

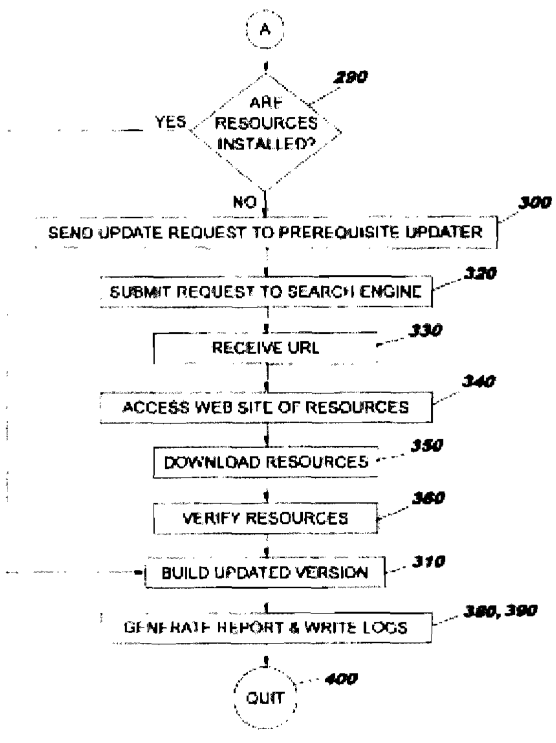
> The updater component then performs on the local computer system a comparison **250** between the current installed software product's identifier and release number and the listed available updates in the retrieved file **160.** This comparison

determines possible growth paths from the current to updated versions, but these possible growth paths are then compared **260** with predefined update criteria, and any possible paths which do not satisfy the update criteria are discarded.

Donohue, col. 10, ll. 59-67.

According to the above description, step 250 is a comparison between a release identifier of a currently installed software and the release identifiers of the available updates for that software. When several updates are available, different ways of reaching the various update levels are possible in *Donohue*. *Donohue* calls these ways "growth paths". Step 260 is a comparison of the possible growth paths for a software with some predefined update criteria. Neither of the cited reference numerals and their corresponding descriptions teaches compatibility information about new software and existing set of software. Therefore, the cited sections of *Donohue* fail to teach the referring step as recited in claim 1.

Next, the Examiner cites the following figure reference and corresponding text in *Donohue* as teaching the providing step:



Donohue, Figure 4B.

The Examiner incorrectly believes that reference numeral 310 in this figure teaches the providing step as recited in claim 1. In this figure, *Donohue* presents further steps in the sequence of operation of the updater component. The steps cited by the Examiner are described in *Donohue* as follows:

> If all required resources are available locally (or on another machine in the case of software relying on some prerequisite software operating on a remote machine), and have been verified, then the updater component progresses to the step **310** (see FIG. **4**) of building the updated software version.

Donohue, col. 11, ll. 56-61.

According to the above description, step 310 is a building step to build the update software version. This teaching in *Donohue* does not teach at least the "providing the <u>compatibility information</u>", "the compatibility information", "used to determine <u>whether to install</u>", and "whether to install <u>the new software</u>" as recited in the providing step of claim 1. Step 310 is simply a build step, that is, preparing a software update for installation. *Donohue* does not provide for a compatibility information, or a decision whether to install even the update, much less a new software. Therefore, the cited sections of *Donohue* fail to teach the providing step as recited in claim 1.

Therefore, *Donohue* does not anticipate claims 1-2 under 35 U.S.C. § 102(b). Furthermore, independent claims 3, 8, 13, and 18 contain at least some features that are present in claim 1. Therefore, *Donohue* does not anticipate independent claims 3, 8, 13, and 18 either. Consequently, dependent claims 4-5, 9-10, 14-15, and 19-20 are also not anticipated at least by virtue of their dependence from one of these independent claims.

The Examiner has rejected independent claim 6, 11, 16, and 19 on basis similar to the rejection of one of the above independent and dependent claims, without additional citations or reasoning specific to those claims. Therefore, for the reasons described above, *Donohue* further fails to anticipate independent claims 6, 11, 16, and 19 under 35 U.S.C. § 102(b). Dependent claims 7, 12, and 17 are also not anticipated by *Donohue* at least by virtue of their dependence from one of these independent claims.

Dependent claims contain additional features not taught by *Donohue*. For example, *Donohue* does not teach the testing step as recited in claim 2. The Examiner collectively cites *Donohue's* reference numeral 20 in Figure 1, 260 in Figure 4A, and 360-310 in Figure 4B together with their associated text as teaching all the steps of claim 2. The Examiner states:

> The rejection of base claim 1 is incorporated. Donohue further teaches responsive to a first selected user input, testing the new software module in a test data processing system in combination with the existing set of software modules; and responsive to a second selected user input, installing the new software module in the data processing system (see at least 20 FIG.1 & associated text; 260 FIG.4A & associated text; 360-310 FIG.4B & associated text).

Office Action dated December 19, 2006, p. 3.

A detailed description has already been provided for reference numerals 310, 260 above. Additionally, reference numeral 20 is an updater component as described in the same section of *Donohue* as quoted above to describe reference numeral 10. Reference numeral 360 is described in *Donohue* as follows:

> After verifying 360 the legitimacy of the down loaded resources, the updater component automatically builds 310 the installation in the target environment in accordance with the update policy.

Donohue, col. 12, ll. 35-38.

According to this description, reference numeral 360 is a verifying step that verifies the legitimacy of the downloaded resource, such as an update file. Collectively, reference numerals 20, 260, 310 and 360 fail to teach testing of anything, much less the testing step as recited in claim 2. Additional steps in *Donohue's* figure 4B also fail to add anything that may teach the testing step as recited in claim 2. Therefore, *Donohue* fails to anticipate claim 2 for this additional reason.

IV. Conclusion

It is respectfully urged that the subject application is patentable over *Donohue,* and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 19, 2007

Respectfully submitted,

/Rakesh Garg/

Rakesh Garg

Reg. No. 57,434
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicants